Cloud Computing에 대한 조사
- 구조와 경제성에 대하여 -

- Cloud Computing의 시작
: 기업은 자신들의 서비스를 인터넷에 배포하기 위해 데이터센터를 구축하고 있다. 대규모 데이터센터에는 수많은 서버들이 설비되어있다. 대규모 데이터센터에서는 자신들의 서비스를 안정적으로 고객에게 제공하기 위해 실제 필요한 자원보다 더 여유있는 자원을 보유한다. 즉, 특정시기(트레픽이 많이 발생하는)를 제외한 시기에 잉여자원이 발생한다는 것이다. 이러한 잉여자원을 효율적으로 활용하기 위해 이것을 인터넷상의 공유자원으로 전환하여 공유된 자원을 고객(서버를 운영하기 위한 컴퓨팅 자원이 부족한)에게 제공한다. 그 사용분만큼 요금을 부과하는 것으로 잉여자원에 대한 이윤을 창출하려는 아이디어가 Cloud Computing의 시작이다.
- Cloud Computing이란?
: 위에서 언급한 것처럼 클라우드 컴퓨팅이란 "클라우드 제공자가 보유하고 있는 컴퓨팅 자원을 인터넷상의 공유자원으로 전환하여 그것을 고객에게 제공함으로써 이윤을 창출하는 것"이다.
- Cloud Computing의 필요성
: 웹상에 자신이 어떠한 서비스를 제공하고자 한다면 그것을 제공하기 위한 서버가 필요하다. 작은 규모의 서비스(트레픽이 적고, 많은 스토리지, DB를 요구하지 않는)를 제공하는 기업에서는 관리실에 서버컴퓨터를 몇 대 배치하고 그것을 관리하면 된다. 그러나 큰 규모의 서비스를 제공하는 기업은 관리실에서 서버 몇 대 관리하는 것으로 트레픽, 요구되는 스토리지 등을 감당할 수 없을 것이다. 클라우드 서비스가 활성화되기 이전에 기업은 자체 데이터센터[1]를 구축하거나, 다른 전문 호스팅 업체에 외주하는 방법을 선택하였다. 그러나 이 두 방법은 모두 막대한 CAPEX(설비투자비용)이 발생하고 또한 OPEX(운영비용)[전력공급, 관리실 환경 조정, 장애복구 등]도 무시할 수 없는 비용이 요구되었다. 그러나 클라우드 서비스는 초기에 발생하는 모든 CAPEX를 OPEX로 전환시켜 주었다.(이후 설명하겠지만 그렇다고 OPEX가 크게 증가하는 것은 아니다.) 또한 사용분만큼의 비용을 요구하기 때문에 사업이 순조롭지 않게 진행되었을 때에 대한 리스크도 감소하였고 이로 인해 많은 스타트업 기업들의 웹 어플리케이션 배포에 대한 진입장벽을 크게 낮춰주었다.
- 자체 데이터센터와 클라우드 서비스
: 기업의 규모가 거대하여 막대한 CAPEX를 충분히 감당할 수 있는 대기업의 경우 보안, 운영비용에 대한 우려로 자체 데이터센터를 구축할 수 있다. 물론 큰 CAPEX를 지불한 만큼 OPEX는 클라우드 서비스를 이용하는 것에 비해 낮게 측정될 것이다. 그러나 문제는 확장성에서 나타난다. 갑자기 서비스에 대한 트레픽이 폭등하는 경우 자체 데이터센터를 운영하는 기업은 그것에 대비할 수 있을만큼의 여분의 자원을 보유하는 비효율적인 방안을 택하거나 서버가 다운되는 실패를 경험해야 한다. 그러나 대형 클라우드 제공자가 보유하고 있는 컴퓨팅자원은 거의 무한하다.(많은 컴퓨팅 자원이 자신들의 사업수단이기 때문에 필요한 만큼의 자원을 요하는 일반 서비스업체와는 비교할 수 없는 것이 당연하다.) 그렇기 때문에 서비스에 대한 트레픽이 순간적으로 폭등하는 경우, 이미 구비되어있는 컴퓨팅자원을 더 끌어와서 사용하면 된다. 또한 그 순간이 지나면 필요이상의 자원은 다시 공유자원으로 환원되어 비용적인 면에서 탁월하다.
- Cloud Computing의 경제성
: Cloud Computing의 필요성에서 언급했듯이 클라우드 서비스는 CAPEX를 OPEX로 전환시켜주지만 OPEX를 크게 증가시키지 않는다. 그 이유는 "규모의 경제성"에 있다. 대형 클라우드 제공자들은 천문학적인 금액을 들여 엄청난 규모의 데이터센터를 구성한다. 이러한 데이터센터는 값싼 전력을 공급받을 수 있는 지역에 위치하며 전력 공급처와의 계약을 통해 더욱 저렴한 비용으로 전력을 이용할 수 있다. 또한 그들은 하드웨어 공급처와 계약을 통해 일반 고객에 비해 저렴한 가격으로 컴퓨팅자원을 확보할 수 있다. 이렇게 확보된 자원을 가상화를 통해 매우 효율적으로 활용하기 때문에 고객에게 저렴한 서비스를 제공할 수 있는 것이다.(아무리 큰 기업의 자체 데이터센터라 할지라도 이를 사업성으로 활용하는 클라우드 제공자의 효율성을 따라가는 것은 불가능할 것이다.)
- 클라우드 서비스의 4대 요소
1. 공유 자원(Shared Resource)
: 클라우드 서비스가 제공하는 컴퓨팅자원은 공유자원을 기반으로 하여 수요자가 바뀌더라도 그 수요량이 매우 감소하지 않는 이상 엄청난 양의 컴퓨팅자원을 보유하더라도 사업성의 문제가 되지 않는다. 공유자원은 클라우드 제공자가 거의 무한에 가까운 컴퓨팅자원을 보유할 수 있는 근거가 된다.
2. 가상화(Virtualization)

---

1) 데이터센터 : 많은 서버 컴퓨터들의 집합

: 가상화란 하나의 물리적인 장비를 가상으로 여러 대의 논리적 장비로 분할하여 하나의 논리적 장비가 마치 물리적 장비처럼 가동되도록 하는 것이다.(Xen, VMware등을 이용하여 하나의 논리적 장비는 각각의 OS를 가질 수 있고 어플리케이션을 가동할 수 있다.) 가상화는 하드웨어의 가동률을 극적으로 높여주는 역할도 한다. 이렇게 분할된 논리적 장비들은 인스턴스(VMI : Virtual Machine Instance)라고 하며, 클라우드 제공자들은 이것을 수요자가 필요로 하는 만큼 제공한다.

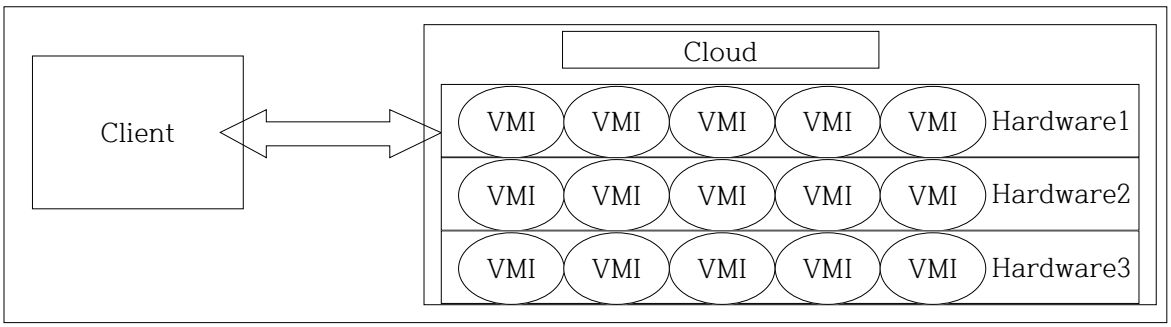

그림 1 Client와 Cloud간의 상호작용

3. 탄력성(Elasticity)과 자동화(Automation)

: 전통적인 서비스와 클라우드 서비스의 가장 큰 차이점이라면 탄력성과 자동화라고 할 수 있다. Amazon의 유명한 EC2는 이름 그대로 Elastic Compute Cloud이다. 그만큼 탄력성은 클라우드 서비스가 내세우는 장점이다. 탄력성이란 클라우드 제공자가 고객에게 컴퓨팅자원을 제공함에 있어서 “필요한 만큼” 제공한다는 것이다. 만약 X-Market이 전자제품 전자상거래 사이트를 운영한다고 하자. 이 사이트는 블랙 프라이데이(전자제품을 매우 저렴하게 판매하는 시점)에 평소보다 훨씬 많은 양의 트레픽을 감당해야 할 것이다.[그림 2] 이 기업이 자체 데이터센터를 운영한다면 블랙 프라이데이의 트레픽을 감당할 만큼의 여분의 컴퓨팅자원(이벤트를 제외한 시기에는 거의 사용되지 않는)을 구비해두어야 할 것이다. 그러나 클라우드 서비스를 이용한다면 많은 트레픽이 몰리는 블랙 프라이데이에 기업은 클라우드로부터 필요한 양의 VMIs를 더 끌어와서 사용하고 트레픽이 줄어들면 다시 이것을 클라우드에 환원함으로써 사용분만큼의 비용을 지불하면 된다. 또한 많은 클라우드 제공자들은 이러한 트레픽에 대한 처리하는데 있어 자동화 기능을 제공한다. 자동화 기능을 이용하면 고객이 직접 트레픽이 몰리는 시점에 더 많은 VMIs를 요구하고 트레픽이 줄어들면 VMIs를 반환하지 않아도 된다. 즉, 클라우드에서 고객의 트레픽을 체크하여 한계 이상의 트레픽이 들어오면 자동으로 인스턴스를 추가하고 일정 시간동안 인스턴스의 평균 사용량이 낮으면 다시 추가된 인스턴스를 클라우드로 반환시켜주는 것이다.(매우 많은 인스턴스가 요구될 경우에는, 고객이 미리 클라우드 제공자에게 어느 시기에 얼마만큼의 인스턴스를 사용하겠다고 말해주어야 한다.)

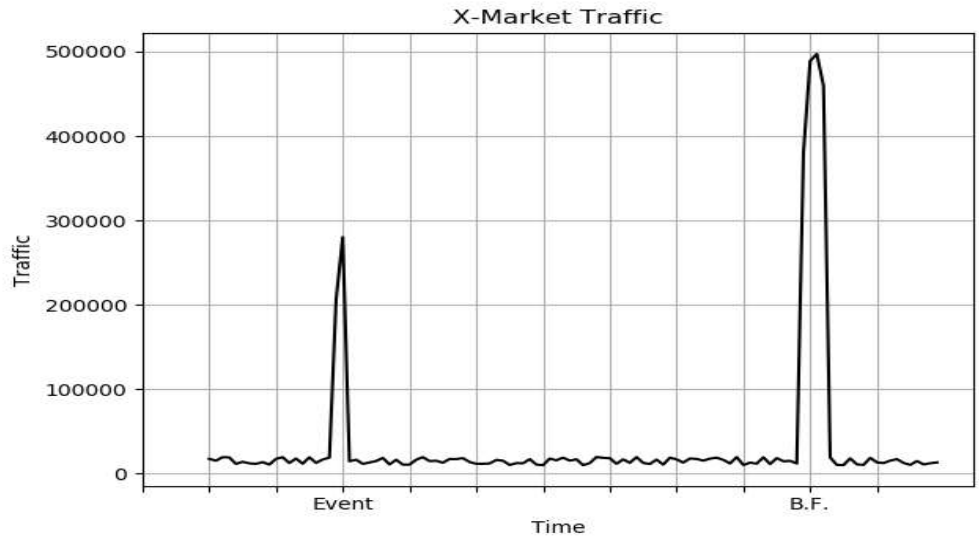

그림 2 기간에 따른 X-Market의 Traffic

4. Pay-as-you-go 방식의 과금

: 클라우드 서비스는 기본적으로 “사용한 만큼 지불”을 원칙으로 한다. 위의 X-Market의 경우 평소에는 낮은 비용으

로 서비스를 제공하다가, Event나 Black-Friday와 같이 많은 인스턴스를 이용한 때에 그것에 대한 비용을 지불하면 된다.(사용자가 많았던 만큼 자신의 이윤 또한 증가했을 것이므로 충분히 그에 대한 비용을 감당할 수 있다.) 이 과금 방식은 Traffic도 적고 이윤도 적은 웹 사이트를 초기 오픈 시점에 아주 합리적인 모델이라고 할 수 있다. 클라우드 제공자들은 이 모델 말고도 계약형 모델 또한 제시하여 특정 기간동안 꾸준한 자원을 이용하면 할인 혜택을 제공한다. 위 X-market의 경우 사이트를 꾸준히 운영할 계획을 가지고 있다면 평소의 트레픽을 감당할 만큼의 인스턴스 정도는 장기 계약을 맺어 단위 비용을 감소시키는 것이 합리적일 것이다.

- Cloud Computing Service의 종류

: 현재 클라우드 제공자가 제공하는 서비스는 매우 다양해졌다. 이것을 크게 3가지로 분류한다면 IaaS, PaaS, SaaS로 나눌 수 있을 것이다. 아래의 [표 1]에서 IaaS, PaaS, SaaS의 차이점을 한 눈에 볼 수 있다.

IaaS	PaaS	SaaS
Application	Application	Application
Platform	Platform	Platform
Web Server	Web Server	Web Server
OS	OS	OS
Virtualized Instance	Virtualized Instance	Virtualized Instance
Hardware	Hardware	Hardware

표 1 IaaS, PaaS, SaaS의 차이점 [녹색 : 사용자 조작 부분, 회색 : 클라우드 제공(고정) 부분]

1. IaaS(Infrastructure as a Service) [서비스형 인프라]

: IaaS는 3가지 모델 중 클라우드 사용자의 자유도가 가장 높은 모델이다. 클라우드 제공자는 가상화된 컴퓨팅자원만을 제공하고, 그것에 OS(Operating System)을 결정하고 어떻게 활용할지는 모두 사용자의 영역이다. 자유도가 높은 만큼 사용자가 조작해야 하는 것이 많다. 대표적인 IaaS의 예로는 AWS EC2, Google Compute Engine, Microsoft Azure Virtual Machines이 있다.

2. PaaS(Platform as a Service) [서비스형 플랫폼]

: PaaS는 클라우드 제공자가 플랫폼까지 제공하는 모델로 개발자가 어플리케이션을 개발하기에 적당한 환경을 클라우드가 미리 조성해서 제공한다. 예를 들어 클라우드 제공자는 Java, Python, Visual Studio 등의 개발 도구들이 설치되어 있는 컴퓨팅자원을 사용자에게 제공한다. 개발자는 이러한 도구를 설치하지 않아도 되므로 개발에 더 집중할 수 있다. PaaS의 대표적인 예로는 Google App Engine, Azure DevOps 등이 있다.

3. SaaS(Software as a Service) [서비스형 소프트웨어]

: SaaS는 클라우드 제공자가 이미 어플리케이션까지 작성하여 사용자는 로그인을 하고 이용하기만 하면 되는 모델이다. 예를 들면 Google Docs, Microsoft Office 365 등과 같이 문서 작성 기능, 엑셀 기능을 인터넷상에서 제공하는 것이다. 이것은 사용자로 하여금 설치에 대한 부담을 줄여줄 뿐만 아니라 라이센스 구매에 대한 부담도 줄여준다.

위에서 살펴본 바와 같이 클라우드는 다양한 범위의 서비스를 제공하여 XaaS라고 불리기도 한다.(X는 장비, 인프라, 플랫폼, 어플리케이션, 데이터센터 등을 모두 포함)

- Cloud Service의 구성 요소

1. 데이터 센터

: 수많은 하드웨어(컴퓨팅자원)들의 집합으로, 인터넷상에서 클라우드 서비스를 제공하는 컴퓨팅자원의 실체가 모여있는 곳이다.

2. 가상화

: 하나의 물리적 자원을 여러 개의 논리적 자원으로 분할하여 장비의 가동률도 높이는 동시에 작은 단위의 인스턴스를 제공함으로써 탄력성도 높여준다.

3. API(Application Programming Interface)

: API는 어플리케이션을 조정하기 위해 사용되는 프로토콜, 도구들의 집합이다. 클라우드 사용자는 클라우드 제공자와 규정된 API를 통해 통신하며 자신의 서비스를 관리/이용한다.

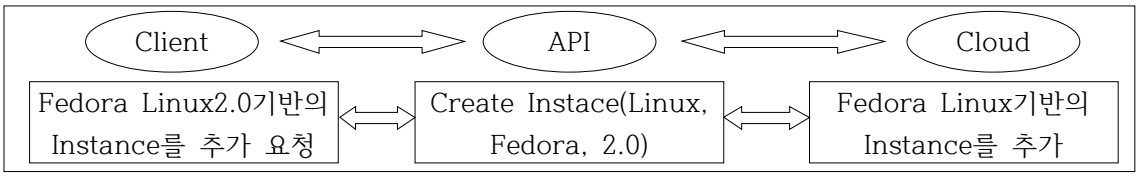

그림 3 API를 통한 Client와 Cloud의 통신

4. Cloud Storage

: 사용자가 서비스에 이용할 파일들을 저장해두는 공간을 클라우드 제공자가 제공하는 것이다. 저장소는 해당 업체의 웹 어플리케이션과 연동성이 좋다.(예를 들면 Amazon EC2는 Amazon S3와 연동성이 좋다.)

5. Database

: 이것은 서비스가 잘 작동하도록 길잡이 역할을 해주는 것이다. 어떠한 서비스에 로그인을 하면 그것이 유효한지를 판단하고, 각 사용자에 맞는 정보를 제공하도록 해준다. 클라우드 서비스에서는 주로 NoSQL DB를 제공한다. NoSQL DB란 비관계형 데이터베이스를 의미한다. 보통의 관계형 데이터베이스들은 많은 테이블을 서로 조인시켜 최대한 데이터의 중복을 피하는 방식을 이용하는데 이것을 클라우드에 적용시키는 것이 쉬운 일이 아니다. 그래서 클라우드는 단순히 [key : value]로 구성된 비관계형 데이터베이스를 제공하는데 이것이 가능한 이유는 클라우드의 디스크 공간 비용이 적기 때문이다. 최근의 클라우드 서비스는 RDBMS를 지원하는 데이터베이스도 제공하고 있다.

6. 탄력성

: 서버 요구량에 따라 가동 인스턴스를 늘렸다 줄였다하는 탄력성은 클라우드의 핵심 구성요소라고 할 수 있다.

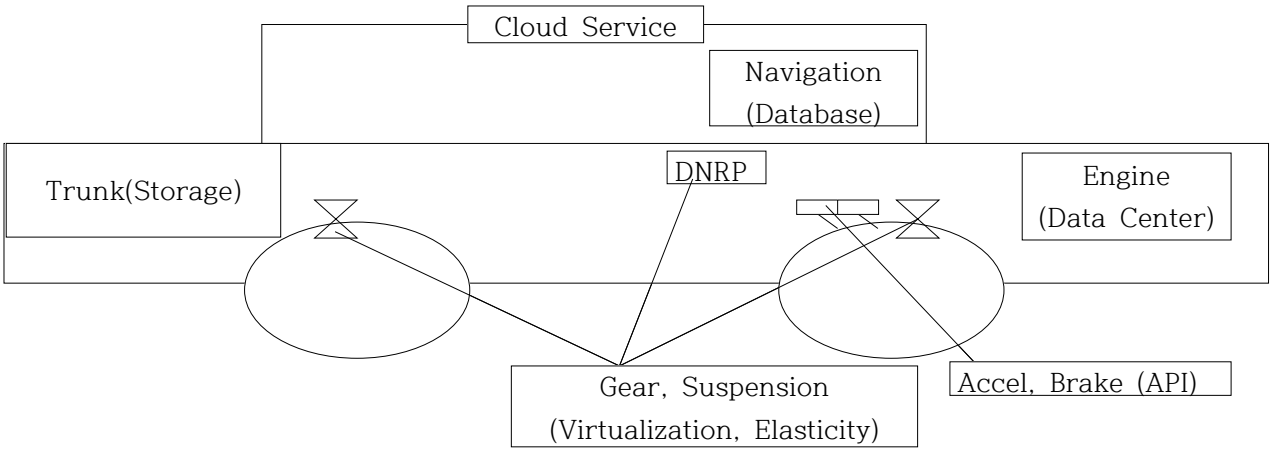

그림 4 자동차에 비유한 클라우드 서비스

- Cloud Service Security [보안]

: 많은 기업들이 클라우드 서비스를 이용하고 있지만, 일부 보안에 큰 관심을 두는 기업은 보안상의 우려로 클라우드 서비스를 꺼린다. 자신들의 극비 문서가 다른 기업의 하드웨어에 저장되어 관리되고 있다는 생각은 당연한 우려이다. 그러나 클라우드 제공자가 악의적으로 그 데이터에 접근하려하지 않는다는 가정 하에 클라우드 서비스를 이용한다는 것은 매우 높은 수준의 보안 혜택을 누리는 것과 같다. 클라우드 제공업체는 SAS 70 인증을 통해 데이터센터의 물리적 보안 수준을 점검받는다. 클라우드 제공자들은 데이터센터의 물리적 접근에 대한 엄중한 관리와 인증을 통해 사용자들로 하여금 보안에 대한 걱정을 덜어준다. 또한 Access Keys, X.509 Certificates, Key Pairs를 통해 인증된 사용자가 아닌 해커들로부터 고객의 정보를 보호하려는 노력을 하고 있다. 엄중한 방화벽의 설계 또한 제공되기 때문에 보안에 이러한 대규모 클라우드 제공업체보다 더 큰 노력을 들이지 않는 이상 자체 데이터센터가 클라우드보다 더 안전하다고 말하기 어렵다. [클라우드 내부의 VMI를 감청모드로 설정해두어도 Traffic 자체가 클라우드의 Hypervisor에 의해 관리되기 때문에 Sniffing, Spoofing 등의 공격에 대한 대비도 되어있다.]

- 분산을 통한 데이터의 병렬 처리

: 클라우드 제공자는 가용자원을 최대한으로 활용하여 서비스의 품질을 향상시키기 위해 많은 노력을 기울이고 있다. 분산을 통한 데이터의 병렬 처리 또한 그 노력들 중 하나이다. 이러한 분산 기능은 연산 처리, DB Query 등에 이용된다. 이것의 기본 아이디어는 하나의 문제를 여러 개로 분할하여 멀티 코어 전체에 배분, 코어 전체를 활용하여 처리된 문제를 다시 합산하여 결과를 도출하는 것이다. 이 분산 기능을 DB에 접목시킨 것이 Sharding을 통한 데이터베이스의 분할이다. Sharding이란 하나의 데이터베이스에 저장되어 있는 많은 정보를 여러 개의 작은 단위(Shard)로 쪼개

서 독립적으로 관리하는 것이다. Sharding은 정보검색에 대한 속도를 향상시키고, 동시에 높은 확장성을 가지게 했을 뿐만 아니라 가용성 또한 높여주었다. 정보검색에 대한 속도가 향상된 이유는 독립적으로 관리되는 Shard들이 자신에 관련된 정보에 대해서만 반응하기 때문이다. 당연히 하나의 큰 데이터베이스에서 정보를 찾는 것보다 작은 데이터베이스에서 정보를 찾는 것이 빠르다. Sharding을 하지않고 하나의 데이터베이스를 사용할 경우, DB의 성능을 향상시키기 위해서는 고가의 장비를 구매하는 방법을 이용하였다. 비용도 많이 들고 성능의 큰 향상은 기대할 수 없는 방법이다. 그러나 Sharding을 통해 작은 Shard들을 관리한다면 DB의 성능 향상을 위해 저가의 DB를 더 늘려주면 된다. 이때 관계형 데이터베이스 모델을 활용할 경우 단순히 DB의 추가 설치로 확장이 어렵기 때문에 클라우드에서 NoSQL DB 모델을 이용하는 것이다. 또한 하나의 DB에서 모든 정보를 관리한다면 그 DB가 장애를 일으키면 모든 서비스가 마비되는 것에 비해 분할하여 관리하는 Sharding의 경우 하나의 DB가 마비되어도 나머지는 정상적으로 작동하기 때문에 가용성이 높다고 할 수 있다. 이렇게 DB를 분할할 때는 어떠한 기준을 가지고 정보를 분할 저장하게 되는데, 이 분할 기준을 잘 정해주어야 각 Shard 정보량의 차이가 발생하지 않고 확장에 문제가 되지 않는다.

- Cloud Computing Stability [안정성]

: 클라우드 서비스는 각 분산 처리 시스템에 대한 낮은 상호의존도를 지향하여 안정성을 높이려는 노력을 하고 있다. 낮은 상호의존성과 대규모 분산 컴퓨팅을 이용의 대표적인 예로는 Map-Reduce가 있다. Map-Reduce는 웹상의 대용량 검색 문제에 대한 해결책으로 제시된 것이다. 이것은 하나의 input을 여러 개의 sub-input으로 Spliting하여 각 sub-input을 Mapper로 지정된 인스턴스들에 부여, Mapper는 자신의 Mapping 함수로 sub-input을 정리한다. 매퍼들에 의해 매핑된 자료들은 Shuffling을 통해 Key값이 같은 것들끼리 모은다. 그렇게 모인 자료를 Reducer로 지정된 인스턴스들로 부여되어 Reducer는 그 value를 합산한다. 이렇게 도출된 sub-output을 합쳐서 하나의 Output을 도출해낸다. 이러한 Map-Reduce를 사용하지 않고 하나의 인스턴스가 해결하려면 매우 긴 시간이 걸리는 문제를 여러 개의 인스턴스들의 병렬 처리로 고가의 장비를 활용하지 않고도 엄청난 속도의 향상을 누리게 된 것이다. 실제로 이 과정에서 사용되는 매퍼, 리듀서로 동작하는 장비들은 베어본 장비[2]들이다. 그렇다면 이러한 저가의 장비를 사용하여 안정성을 높인다는 것은 무슨 말인가? 여기서 낮은 상호의존성이 이용된다. 각 매퍼, 리듀서들은 상호의존도가 낮기 때문에 다른 장비가 마비되어도 자신의 작업에 영향을 끼치지 않는다. 그렇기 때문에 분산 시스템에서는 마비된 인스턴스가 생기면 그 인스턴스의 작업을 다른 인스턴스로 부가하여 전체적인 작업에는 영향이 가지 않도록 한다. 이것으로 인해 분산 처리 시스템은 저가의 장비들로 높은 안정성을 확보한 것이다.

<table>
<tr><th>INPUT</th><th>Spliting</th><th>Mapping</th><th>Shuffling</th><th>Reducing</th><th>OUTPUT</th></tr>
<tr><td rowspan="4">AAAB
CBBA
CBCC
CACB</td><td rowspan="2">AAAB</td><td rowspan="2">A : 3
B : 1</td><td rowspan="2">A : 3
A : 1</td><td rowspan="2">A : 5</td><td rowspan="4">A : 5
B : 5
C : 6</td></tr>
<tr><td rowspan="2">CBBA</td><td rowspan="2">A : 1
B : 2
C : 1</td></tr>
<tr><td rowspan="2">A : 1</td><td rowspan="2">B : 1
B : 2
B : 1
B : 1</td><td rowspan="2">B : 5</td></tr>
<tr><td>CBCC</td><td>B : 1
C : 3</td></tr>
<tr><td></td><td>CACB</td><td>A : 1
B : 1
C : 2</td><td>C : 1
C : 3
C : 2</td><td>C : 6</td><td></td></tr>
</table>

표 2 Map-Reduce를 이용한 분산 처리 시스템 동작 원리

- Cloud Computing에 대한 개인적인 생각

: 미처 기술하지 못한 클라우스 버스팅, 가상 폐쇄형 클라우드, Selenium을 통한 웹 어플리케이션 테스트 자동화 기능, 부하 테스트 등 클라우드 서비스는 5페이지에는 다 담지 못할 정도로 많은 서비스를 제공하고 있었습니다. 저의 개인적인 견해로는 클라우드 컴퓨팅은 향후 많은 스타트업 기업의 필수 코스가 될 것이고 대기업 또한 조금씩 자신의 서비스를 클라우드로 옮겨갈 것으로 생각됩니다.

- REFERENCE

클라우드 세상 속으로(The Cloud at Your Service), 조시 로젠버그 외 1인, 에이콘 출판사, 2015.12.24

2) Bare-bone 장비 : 메인보드에 필요한 장치만 연결하여 사용하는 저가의 장비